



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/654,115	08/30/2000	Dean A. Klein	500050.01	5616

27076 7590 09/28/2004

DORSEY & WHITNEY LLP  
INTELLECTUAL PROPERTY DEPARTMENT  
SUITE 3400  
1420 FIFTH AVENUE  
SEATTLE, WA 98101

EXAMINER

VU, TUAN A

ART UNIT

PAPER NUMBER

2124

DATE MAILED: 09/28/2004

10

Please find below and/or attached an Office communication concerning this application or proceeding.

52

# Office Action Summary

Application No.

09/654,115

Applicant(s)

KLEIN, DEAN A.

Examiner

Tuan A Vu

Art Unit

2124

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

## Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

## Status

- 1) ☒ Responsive to communication(s) filed on 19 December 2003.
- 2a) ☐ This action is FINAL. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

## Disposition of Claims

- 4) ☒ Claim(s) 1-61 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 1-61 is/are rejected.
- 7) ☐ Claim(s) \_\_\_\_\_ is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

## Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on \_\_\_\_\_ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

## Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some \* c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
2. ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).
- \* See the attached detailed Office action for a list of the certified copies not received.

## Attachment(s)

- 1) ☒ Notice of References Cited (PTO-892)
- 2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)  
Paper No(s)/Mail Date \_\_\_\_\_.
- 4) ☐ Interview Summary (PTO-413)  
Paper No(s)/Mail Date \_\_\_\_\_.
- 5) ☐ Notice of Informal Patent Application (PTO-152)
- 6) ☐ Other: \_\_\_\_\_.

### DETAILED ACTION

1. This action is responsive to the Applicant's response filed 12/19/2003.

As indicated in Applicant's response, no claims have been amended. Claims 1-61 are pending in the office action.

Regarding the above response, Applicant's request for reconsideration of the finality of the rejection of the last Office action is persuasive and, therefore, the finality of that action is withdrawn.

### *Claim Rejections - 35 USC § 102*

2. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

Note: 35 U.S.C. § 102(e), as revised by the AIPA and H.R. 2215, applies to all qualifying references, except when the reference is a U.S. patent resulting directly or indirectly from an international application filed before November 29, 2000. For such patents, the prior art date is determined under 35 U.S.C. § 102(e) as it existed prior to the amendment by the AIPA (pre-AIPA 35 U.S.C. § 102(e)).

3. Claims 25-27, 29, 32-36, 38-40, 46-50, 52-54 and 60-61 are rejected under 35 U.S.C. 102(e) as being anticipated by Mattson, Jr. et al., USPN: 6,115,809 ( hereinafter Mattson).

As per claim 25, Mattson discloses a system for determining which portions of a program code to cache and which to not cache, comprising:

a memory device containing a program code (e.g. col. 5, lines 25-56; see col. 3, lines 46-51 or *Holler*, USPN: 5721893, Fig. 1); and

a processor connected to the memory device, the processor being adapted to be controlled by the program code to direct selected portions of the program code to a cache ( e.g. *static code cache*, *dynamic code cache* - Fig. 3) based at least in part on cacheability determinations made during compilation (e.g. *compiler option*, *Holler* , *at compile time*- col. 6, lines 18-45; *once at compile time* – col. 6, lines 59-62 ) of a computer program.

**As per claims 26 and 27**, Mattson discloses that the information comprises instructions (e.g. Fig. 1) and data accessed by the computer program (e.g. Fig. 1 ).

**As per claim 29**, Mattson discloses memory connected to processor via bus circuitry (Note: bus circuitry connecting processor to memory or peripheral device is inherently disclosed – if necessary see col. 3, lines 46-51 or *Holler*, USPN: 5721893, Fig. 1).

**As per claims 32 and 33**, Mattson further discloses a memory device comprising a main memory and an external storage device connected to the processor via the bus (Note: bus circuitry connecting processor to fast memory or peripheral device or main memory is inherently disclosed – if necessary see col. 3, lines 46-51 or *Holler*, USPN: 5721893, Fig. 1).

**As per claim 34**, Mattson discloses a method for controlling the cacheability of information in a computer system, comprising:

compiling a computer program, by making cacheability determination for information associated with the computer program (e.g. e.g. *static code cache*, *dynamic code cache* - Fig. 3; see col. 6, lines 18-33 or *Holler*, Fig. 4); and

marking at least selected portions of the information according to the determinations (e.g. Fig. 4 - Note: profile-based determination, i.e. *profiled as* 'strong' of 'weak' -is equivalent to marking for appropriate cache storage action; *flag* - col. 9, lines 23-46);

executing the computer program on a computer system, the computer system including cache circuitry (Fig. 3);

detecting the marking of the selected portions of the information during execution of the computer program (e.g. col. 9, lines 1-22); and

directing the selected portions of the information to the cache circuitry according to the marking (e.g. col. 9, lines 1-22; 47-51; Fig. 3-4 ).

**As per claims 35 and 36**, refer to rejections 26 and 27 respectively.

**As per claims 38 and 39**, Mattson further discloses compiling comprises translating source code of the computer program into object code; and programming object code into program directly (e.g. col. 6, lines 18-45; Fig. 3- Note: block partitioning in conjunction with branch prediction heuristics --see col. 6, lines 18-33 or *Holler*, Fig. 4-- before gathering profiling data is equivalent to compiling including partitioning and heuristics-based instrumentation, i.e. generate an object code leading to an executable to provide subsequent profile-based optimization)

**As per claim 40**, Mattson further discloses cacheability determinations as to whether the selected portions are cacheable (e.g. Fig. 3).

**As per claim 46**, Mattson further discloses that the cacheability determination is making determination for a piece of information based on frequency that it will be accessed by the processor during execution of the program ( e.g. col. 7, lines 15-44).

**As per claim 47**, Mattson further discloses a compiler to optimize cacheability determination for a piece of information based on what other piece of information is likely to be overwritten if the first piece is cached (e.g. col. 3, lines 46-67 – Note: branch prediction incorporated by reference discloses concern of overwriting a cache entry).

**As per claim 48**, Mattson discloses a method for compiling a computer program, comprising:

making cacheability determinations for information associated with the program (e.g. *static code cache*, *dynamic code cache* - Fig. 3; see col. 6, lines 18-33 or Holler, Fig. 4 ); and

marking at least selected portions of the information according to the determinations (e.g. Fig. 4 - Note: profile-based determination, i.e. *profiled as* ‘strong’ of ‘weak’ -is equivalent to marking for appropriate cache storage action; *flag* - col. 9, lines 23-46).

**As per claims 49 and 50**, see claims 26-27 respectively.

**As per claims 52 and 53**, see rejection of claims 38-39 respectively.

**As per claim 54**, see rejection of claim 40.

**As per claims 60 and 61**, these claims correspond to claims 46 and 47 above, respectively, hence are rejected herein using the corresponding ground of rejection as set forth therein.

### ***Claim Rejections - 35 USC § 103***

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

5. Claims 1-4, 6-8, 13-15, and 17-24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mattson et al., USPN: 6,115,809, and in view of Morrison et al., USPN: 5,765,037 ( hereinafter Morrison).

**As per claim 1**, Mattson discloses a computer system having cache circuitry, the computer system adapted to be controlled by a computer program to cache information, comprising:

cache circuitry, including a cache memory to store computer program information; a main memory ( e.g. col. 5, lines 25-56; see col. 3, lines 46-51 or *Holler*, USPN: 5721893, Fig. 1);

a processor adapted to be controlled by the computer program ( e.g. col. 4, lines 24-35 or see *Shah*, USPN: 6,205,545, col. 4, lines 28-50 – Note: a processor executing a program inherently signifies it being controlled by the program) and to direct selected portions of the information to the cache circuitry based at least in part on cacheability determinations (e.g. Fig. 3) made during compilation of the computer program (e.g. *compiler option, Holler , at compile time*- col. 6, lines 18-45; *once at compile time* – col. 6, lines 59-62 ); and

bus circuitry, connecting the processor, the cache circuitry, and the main memory (see col. 3, lines 46-51 or *Holler*, USPN: 5721893, Fig. 1– Note: bus circuitry is inherent to any computer system).

But Mattson does not explicitly mention directing selected portions of the program information to the cache circuitry via cooperation with a bus interface unit. With the so-disclosed hardware components by Mattson (e.g. see col. 3, lines 46-51 or *Holler*, USPN: 5721893, Fig. 1), one skill in the art would recognize the obvious existence of an interface unit to

Art Unit: 2124

communicate the cache circuitry and the bus system as inferred from Mattson. Further, in a system to route instructions data to the appropriate hardware, e.g. cache, using compilation information (Morrison: col. 33, lines 16-23) analogous to Mattson's system to optimize execution time for branch operations via profiling information prior to caching execution instructions/data, Morrison discloses the use of bus interface unit (unit 1544 – Fig. 15) to cooperate with the execution for translating address and direct instructions to cache (e.g. col. 27, line 56 to col. 28, line 31). Hence, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement a bus interface unit as taught by Morrison to Mattson's system, in case the latter does not already include one such unit, because this would allow the intermediate step of spatially and logically re-directing or selectively dispatching of data to the correct storage place, cache or buffers, enhancing fault-free transmission of cacheable data via bus ( Morrison: col. 30, lines 4-17).

**As per claims 2 and 3**, these claims correspond to claims 26 and 27 above, respectively, hence are rejected herein using the corresponding ground of rejection as set forth therein.

**As per claim 4**, Mattson further discloses that selected portions are marked during the compilation of the program such that the bus interface unit can identify the selected portions during execution of the program (e.g. col. 6, lines 18-45; Fig. 3- Note: block partitioning in conjunction with branch prediction heuristics --see col. 6, lines 18-33 or Holler, Fig. 4-- for profiling data based on compiler instrumentation or rule-based heuristics is equivalent to marking during compilation for information capture leading to identifying of marked portions for cache storage during the execution of the optimized and recompiled program – see col. 9, 1-51; the identifying by the BIU being obvious as per the rationale as set forth in claim 1).



**As per claims 6 and 7**, these claims correspond to claims 38 and 39 above, respectively, hence are rejected herein using the corresponding ground of rejection as set forth therein.

**As per claim 8**, Mattson further discloses that cacheability determinations comprise determining that the selected portions are cacheable (e.g. Fig. 3; see col. 6, lines 18-33 or Holler, Fig. 4).

**As per claim 13**, Mattson further discloses a compiler to optimize cacheability determination (e.g. col. 6, lines 18-45; Fig. 3- Note: block partitioning in conjunction with branch prediction heuristics --see col. 6, lines 18-33 or Holler, Fig. 4-- for gathering profiling data is equivalent to compiler operable for including partitioning and heuristics-based instrumentation leading to profile-based determining on cacheability of identified portions in the final optimized code).

**As per claim 14**, Mattson discloses cacheability determination for a first piece of information is based at least in part on whether caching of the first piece of information is likely not to cause cache mis-prediction or potential cache miss, all of which under the scheme of seeking improved execution efficiency and cache management ( e.g. col. 1, lines 41 to col. 3, line 67) but does not specify whether such caching is likely to cause thrashing of the cache circuitry. The problem of cache being limited leading subsequent techniques to manage cache resources to avoid thrashing was a known concept in the art at the time the invention was made. Morrison for example, in the system to optimize the execution of programs using a bus and cache circuitry associated with profile information analogous to that of Mattson, discloses parameters (*tag*, *IFT*, *LPN*- col. 31, lines 24-64) based on which to process execution and prevent cache miss (e.g. unit 1640-Fig. 16; col. 32, lines 16-44 – Note: cache miss avoidance is equivalent to minimizing

Art Unit: 2124

cache thrashing). It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement parameters determined at compile as taught by Mattson in order to prevent cache thrashing issue as suggested by Morrison because the intent to improve processor latency as suggested by Mattson necessarily implies alleviating memory issues such as cache misses and recovery, i.e. thrashing, which entails processing downtime.

**As per claim 15**, Mattson does not explicitly specify a cache management scheme but discloses a scheme to prevent cache mis-prediction or potential cache miss, all of which under the scheme of seeking improved execution efficiency or cache management ( e.g. col. 1, lines 41 to col. 3, line 67) while Morrison suggests a management scheme in association with the hardware used to process instruction stream, bus and cache circuitry and an cache control structure(e.g. manage -col. 6, lines 1-22; Fig 15-16). It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement a cache management scheme in the cache and I/O circuitry as suggested by Morrison to Mattson's system to process instruction data during execution and base the cacheability determination thereupon because the more complicated the system is in terms of bus size, memory and instructions set architecture, the more management is needed in coordinating data from their source storage, e.g. memory, registers, to their destination, e.g. cache, via bus circuitry; such coordination being routing, and/or synchronization of instructions stream as mentioned by Morrison.

**As per claim 17**, Mattson further discloses a compiler to optimize cacheability determination for a piece of information based on frequency that it will be accessed by the processor at execution (e.g. col. 7, lines 11-44 ).

Art Unit: 2124

**As per claim 18**, Mattson further discloses a compiler to optimize cacheability determination for a piece of information based on what other piece of information is likely to be overwritten if the first piece is cached (e.g. col. 3, lines 46-67 – Note: branch prediction incorporated by reference discloses concern of overwriting a cache entry).

**As per claim 19**, Mattson further discloses that determinations are accomplished during compilation into object code utilizing profile-based optimizations ( e.g. col. 6, lines 21-45).

**As per claims 20 and 21**, Mattson discloses a system controller, adapted to send and retrieve instructions; and a bus device connecting an external device to the bus ( refer to rejection of claims 32 and 33 from above).

**As per claim 22**, Mattson discloses an external device providing instructions utilized by the processor for optimization (e.g. see col. 3, lines 46-51 or *Holler*, USPN: 5721893, Fig. 1– Note: memory/media storage storing profile data is equivalent to providing stored instructions for optimization/execution).

**As per claims 23 and 24**, Mattson further discloses that instructions are compiled by a compiler to optimize cacheability determinations (e.g. . *compiler option, Holler , at compile time*- col. 6, lines 18-45; *once at compile time* – col. 6, lines 59-62; Fig. 3- Note: block partitioning in conjunction with branch prediction heuristics --see col. 6, lines 18-33 or *Holler*, Fig. 4-- before gathering profiling data is equivalent to compiling including partitioning and heuristics-based instrumentation for cacheability determination) and that cache circuitry and processor are provided on one single chip (col. 3, lines 46-51 or *Holler*, USPN: 5721893, Fig. 1).

Art Unit: 2124

6. Claims 5, 12, and 16 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mattson et al., USPN: 6,115,809, and Morrison et al., USPN: 5,765,037, as applied to claims 1, 15 above, and further in view of Prasanna, USPN: 6,272,599 ( hereinafter Prasanna).

As per claim 5, Mattson discloses using compiler including analysis information to determine cacheability of program data and marking of instructions for appropriate storing in cache (Fig. 3-4), or selectively caching during execution based on compiling information generated from partitioning or heuristics-based instrumenting of instructions (see col. 6, lines 18-33 or Holler, Fig. 4); but fails to disclose that such compiler-generated information contains marking bits, and that the compiler sets the marking bits of the selected portions of the information during compilation. As another approach to Mattson's compiler-based insertion of instrumentation instructions as mentioned above, Prasanna, in a system to improve execution time by alleviating cache thrashing using compilation information analogous to the compile time determining of instructions caching as taught by Mattson, discloses the compiler marking of bits in instructions datum, e.g. ILP, in order to identify which lines to cache or not to cache in order to avoid interferences patterns ( *cache/no-cache bit* – col. 2, lines 29-58; Fig. 1-2). It would have been obvious for one of ordinary skill in the art at the time the invention was made to add the use of marking bits to instructions datum by the compiler as suggested by Prasanna to Mattson 's method of determining of target address for execution data access optimization because this would further speed up the translating of address information into cached data in the execution of the optimized code as taught by Mattson, to thereby reduce cache thrashing as also suggested by Prasanna (col. 1, lines 48-67).

**As per claim 12**, Mattson combined with Morrison does not specify that the cache-circuitry includes at least one N-way associative cache with  $N > 1$ ; but Morrison suggests a speed increase in fetching instructions from cache should a fully-associative memory be used ( e.g. col. 31, lines 24-63). Further, Prasanna in the system to mark bits in instruction datum at compile time for cacheability determination as mentioned above, discloses a N-way associative cache ( col. 3, lines 1-14). It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the cache circuitry in the memory latency optimization scheme as taught by the combination Morrison/Mattson in a way that a more-than-one-way associative type of cache as taught by Prasanna because according to Prasanna, this would improve performance of a computer when the size of the sequential address stream to process is greater than the total cache memory available.

**As per claim 16**, Mattson in view of Morrison discloses a cache management scheme as addressed in claim 5 above; but fails to disclose such cache management scheme comprises the level of associativity of cache memory. But in view of Prasanna in the disclosing of a N-associative cache scheme as mentioned in claim 12 above, it would have been obvious for one of ordinary skill in the art at the time the invention was made to incorporate in the cache management scheme as suggested by Mattson/Morrison the level of associativity of cache as taught by Prasanna because of the same benefit mentioned above in the rejection set forth in claim 12.

7. Claims 9-11 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mattson et al., USPN: 6,115,809, and Morrison et al., USPN: 5,765,037, as applied to claim 1 above, and further in view of Lasserre, US Pub. No: 2002/0078268 ( hereinafter Lasserre).

As per claim 9, Mattson discloses cacheability determinations as to whether to cache selected portions in the program but does not specify that the cache circuitry includes first cache memory and second cache memory; nor does Mattson disclose said determinations comprise determinations as to cache said selected portions in the first or second cache memory. However, Mattson suggests 2 types of cache, a dynamic type for hard to predict branch instructions and a static type for easy to predict instructions (Fig. 3,5), hence has disclosed a cache level where fast fetching therefrom can be done and a cache level where slower fetching is likely. Morrison, in a system as mentioned in claim 1 to cache data using compilation information analogous to Mattson's method, discloses the several partitions of cache to store instructions ( e.g. Fig. 15; col. 28, lines 15-31). Further, Lasserre, in a system analogous to Mattson and Morrison' s, using cache circuitry and pre-configured memory information to determine dynamic cache storage to alleviate cache conflicts, discloses level 1 and level 2 caches with different size and application each ( p. 3, paragraphs 0048, 0050; cache 113, 114-Fig. 1). In view of the teachings and suggestions by Mattson/Morrison and Lasserre's suggestion of different size caches and purposes, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement 2 level caches as suggested therefrom and combine the cache determination and partitioning as suggested by Mattson (and further enhanced by Morrison) with the use of 2 level caches to determine as to which cache the portions selected by compilation in Mattson's method should be stored. One of ordinary skill in the art would be motivated to do so because directing a certain size instruction or data to be cached in an appropriate size cache partition as suggested by Lasserre would enhance resource usage efficiency and also in terms of memory spatial and temporal optimization.

**As per claim 10**, Mattson does not teach the first level cache and the second level cache; but in view of Morrison and Lasserre's teachings in claim 9 above, this limitation would have been obvious herein using the same rationale set forth therein.

**As per claim 11**, Mattson does not specify that the cache circuitry supports write-back and write-through caching and that cache determinations comprise the write-back or write-through method. Official notice is taken that in techniques used for cache management such as taught by Mattson, the write-back and write-through techniques were well-known practices at the time the invention was made, so as to avert runtime memory from being bogged down with extraneous read and write operations. Lasserre, in a system to alleviate cache conflict using memory information to set cacheability for execution, discloses both write-back and write-through caching methods (e.g. p. 6, paragraph 0086). It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement a cache system as suggested by Lasserre to that disclosed by Mattson (and further enhanced by Morrison) because these caching methods are known to support selective reduction of latency with regard to the temporary or stable state of the data being used during execution, thus enhancing further execution time efficiency.

8. Claims 28, 37, 45, and 51 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mattson et al., USPN: 6,115,809, as applied to claim 25, 34, 48 above, in view of Prasanna, USPN: 6,272,599.

**As per claim 28**, Mattson discloses using compilation analysis information to determine cacheability of address data (Fig. 3-4), or selectively caching during execution based on compiling information generated from partitioning or heuristics-based instrumenting of

instructions (see col. 6, lines 18-33 or *Holler*, Fig. 4); but fails to disclose that such information contains marking bits, and that the compiler sets the marking bits of the selected portions of the information during compilation. Prasanna, in a system to improve execution time by alleviating cache thrashing as mentioned in claim 5 above, discloses the compiler marking of bits in instructions datum, e.g ILP, in order to identify which lines to cache or not to cache in order to avoid interferences patterns (*cache/no-cache bit* – col. 2, lines 29-58; Fig. 1-2). It would have been obvious for one of ordinary skill in the art at the time the invention was made to add the use of marking bits to instructions datum by the compiler as suggested by Prasanna to Mattson 's method of determining of target address for execution data access optimization for the same rationale as set forth in the rejection of claim 5 above.

**As per claim 37**, Mattson fails to disclose that each piece of information contains marking bits and that such marking includes setting the marking bits of at least the selected portions of the information. In view of the marking of bits in the instructions datum as taught by Prasanna in claim 28 or 5 above, this limitation would also have been obvious and is herein rejected with the same rationale as set forth in the rejection of claim 5 above.

**As per claim 45**, Mattson discloses at least one level of cache memory ( Fig. 3) but fails to disclose that the act of making cacheability determinations for program information is based on a level of associativity of cache memory. But in view of Prasanna in the disclosing of a N-associative cache scheme as mentioned in claim 12 above, it would have been obvious for one of ordinary skill in the art at the time the invention was made to incorporate in the cache circuitry as suggested by Mattson at least one level of associativity of cache as taught by Prasanna because of the same benefit mentioned above in the rejection set forth in claim 12.



Art Unit: 2124

**As per claim 51**, this claim includes the same limitation as claim 37 above, hence is rejected herein using the same rationale as set forth therein.

9. Claims 30-31, 41-44, and 55-58 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mattson et al., USPN: 6,115,809, as applied to claim 25, 34, and 48 above, in view of Lasserre, US Pub. No: 2002/0078268.

**As per claims 30 and 31**, Mattson discloses a cache circuitry connected to processor (Fig. 3) but does not explicitly disclose ( re claim 30) a level one cache; nor does Mattson disclose ( re claim 31) a level two cache connected to the processor and memory via bus circuitry. But in view of the teachings by Lasserre as mentioned in claim 9 above (p. 3, paragraphs 0048, 0050; cache 113, 114-Fig. 1), these level one and level two limitations would have been obvious for the same rationale as set forth in claim 9 above.

**As per claim 41**, Mattson does not explicitly disclose that the cache circuitry includes first cache memory and second cache memory; nor does Mattson disclose said determinations comprise determinations as to cache said selected portions in the first or second cache memory. However, Mattson suggests 2 types of cache, a dynamic type for hard to predict branch instructions, and a static type for easy to predict instructions (Fig. 3,5), hence has disclosed a cache level where fast fetching therefrom can be done and a cache level where slower fetching is likely. Lasserre's use of 2 level caches (p. 3, paragraphs 0048, 0050; cache 113, 114-Fig. 1) has been mentioned in claim 9 above. Hence, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement 2 level caches as suggested by Lasserre and combine the cache determination as suggested by Mattson (and further enhanced by Morrison) with those 2 level caches to determine as to which cache level the portions selected by

Art Unit: 2124

compilation in Mattson's method should be stored. One of ordinary skill in the art would be motivated to do so because of the same benefits as set forth in rejection of claim 9 above.

**As per claim 42**, Mattson does not specify both write-back and write-through caching methods, nor does Mattson disclose determining whether to cache selected portions using those 2 methods. This claim limitations have been addressed using Lasserre's teachings (e.g. p. 6, paragraph 0086) in claim 11 above, hence is rejected herein using the same grounds of rejection as set forth therein.

**As per claim 43**, Mattson discloses cacheability determination for a first piece of information is based at least in part on whether caching of the first piece of information is likely not to cause processor to have a mis-prediction conflict (e.g. e.g. col. 1, lines 41 to col. 3, line 67), but does not specify whether such caching is likely to cause thrashing of the cache circuitry. The problem of cache being limited leading subsequent techniques to manage cache resources to avoid thrashing was a known concept in the art at the time the invention was made. Lasserre, in a system to alleviate cache conflict using memory information to set cacheability of data for execution analogous to that of Mattson, discloses cache circuitry handling cache miss/hit logic and clean-up process as a result of cache miss (e.g. *Hit/Miss logic 510* - Fig. 4; p. 4, 5, 6 - paragraphs 0062, 0063, 0074). By cache miss handling and recovering, one skilled in the art would recognize the cache thrashing issue as claimed. In view of the above known concepts and recognitions, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement information determined at compile as taught by Mattson in order to prevent cache thrashing issue as suggested by Lasserre because the intent to improve

Art Unit: 2124

processor latency as suggested by Mattson necessarily implies alleviating memory issues such as cache misses and recovery, i.e. thrashing, which entails processor downtime.

**As per claim 44**, Mattson does not specify a cache management scheme but discloses a scheme to store instruction data in memory and an I/O controller (e.g. col. 3, lines 46-51 or *Holler*, USPN: 5721893, Fig. 1) while Lasserre suggests a management of cache in association with the hardware used to process instruction stream (e.g. *Hit/Miss logic 510*, *cache control 530* - Fig. 4; p.6 – paragraph 0074). It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement a cache management scheme via the cache control and hit/miss structure as suggested by Lasserre to Mattson's system to process instruction data during execution and base the cacheability determination thereupon because the more complicated the system is in terms of bus size, memory and instructions set architecture, the more management is needed in coordinating data from their source to their destination; such coordination and flow control for optimizing the performance on specialized processors instructions set with numeric intensive execution as suggested by Lasserre ( col. 1, paragraphs 0004, 0005).

**As per claim 55**, Mattson does not disclose that cacheability determinations comprise determinations as to cache said selected portions in the first or second cache memory. But in view of the teachings of Lasserre as mentioned in claim 41 above, this claim is rejected using the same rationale as set forth therein.

**As per claim 56**, this claim includes the limitation as whether to cache using write-back or write-through method as in claim 42 above, hence is rejected using the corresponding rationale set forth therein.

Art Unit: 2124

**As per claims 57 and 58**, these claims correspond to claims 43 and 44 above, respectively, hence are rejected herein using the corresponding ground of rejection as set forth therein.

10. Claim 59 is rejected under 35 U.S.C. 103(a) as being unpatentable over Mattson et al., USPN: 6,115,809, and Lasserre, US Pub. No: 2002/0078268, as applied to claim 58 above, and further in view of Prasanna, USPN: 6,272,599.

**As per claim 59**, Mattson combined with the teachings of Lasserre, fails to disclose that the cache management scheme comprises the level of associativity of the cache. But Prasanna, in the invention as mentioned in claim 45 above, discloses a N-way associative cache memory. In combination with the cache circuitry and cache management scheme of Mattson/Lasserre (re claim 44), Prasanna's teachings would have help render the above limitation obvious for the same rationale as set forth in claim 45 above.

#### ***Response to Arguments***

11. Applicant's arguments with respect to claims 1-61 have been considered but are moot in view of the new ground(s) of rejection.

#### ***Conclusion***

12. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (703)305-7207. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (703)305-9662.

**Any response to this action should be mailed to:**

Art Unit: 2124

Commissioner of Patents and Trademarks

Washington, D.C. 20231

**or faxed to:**

(703) 872-9306 ( for formal communications intended for entry)

**or:** (703) 746-8734 ( for informal or draft communications, please consult Examiner before using this number)

Hand-delivered responses should be brought to Crystal Park II, 2121 Crystal Drive, Arlington. VA. , 22202. 4<sup>th</sup> Floor( Receptionist).

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

VAT  
September 9, 2004

  
ANIL KHATRI  
PRIMARY EXAMINER